Kernel Testing Frameworks

OOSC 2025
September 6 2025

Shuah Khan
Kernel Maintainer & Linux Fellow
The Linux Foundation
skhan@linuxfoundation.org
ShuahKhan

We can all agree that it is important to test the kernel. There are a number of tools for code coverage, dynamic and static analysis to choose from to test the kernel. Linux kernel provides two testing frameworks to write and run tests. A majority of kernel tests are written using these two frameworks. Having a rich set of tools and frameworks is great. However it can be challenging to know which one to chose.

In this talk, Shuah will give an overview of tools and frameworks and their differences, and how to use them to achieve the your kernel testing objectives and goals.

Why kselftest?

- Regression test suite
- Closed and Open Box tests
- Tests kernel from user-space
- User-space applications
  - (Shell scripts, C programs)
  - Kernel Test modules used to exercise kernel code paths
- Allows for breadth and depth coverage (error paths etc.)
- Not for workload or application testing

Why kselftest?

- Perfect for
  - feature, functional and regression testing
  - bug fix focused regression testing and subsystem testing
  - testing user APIs, system calls, critical user paths, common use cases
  - Perfect for end to end regression testing

- Open and Closed box testing – quick assurance that everything works

Kernel Validation With Kselftest

Why KUnit?

- Focuses on in-kernel testing
- Perfect for:
  - testing internal kernel APIs
  - libraries, drivers, …,
  - individual *units* of code
- Perfect for unit testing
  - makes it easy to test all the edge cases
- Openbox tests

KUnit Testing Strategies

Kselftest & KUnit

Kselftest

- Good for depth testing covering deeper code paths
- Good for testing commonly used code paths
- A good test could test some error paths
- Multiarch support

KUnit

- Good for targeting error paths & edge cases
- Easier and faster for zeroing in on a kernel area
- Does KUnit support running on architectures other than UML?
  - Yes.

Kernel Testing Guide

- Writing and Running Tests
- Kselftest vs. KUnit
- Which one to choose?
- How to choose?

Choosing a framework

- What are the goals?
- Regression testing?
- Closed or open box
- Are there any user-space interaction?
- Is it a unit test?
- What about multi-arch support?
- What about code coverage?

How many tests?

- Kselftest
  - 192 individual Makefiles
  - 125 subsystems (directories)
- KUnit
  - Spread around the kernel
  - ./tools/testing/kunit/kunit.py run --alltests --arch x86_64
  - Testing complete. Ran 6701 tests: passed: 6668, skipped: 33
  - Elapsed time: 142.975s total, 4.004s configuring, 85.063s building, 53.774s running

What’s next?

- How do I know which tests to run for my use-case?
  - Kselftests or KUnit - both
- But which tests do I choose to run?

Questions and Discussion

- Is there support for expected fail/pass
  - Yes - References:
  - Linux Kernel Selftests, kselftes.h, kselftest_harness.h
- What happens when dependencies aren't met?
  - Tests run as many test cases as they can
  - Skipping tests is fine grain. Skip just the ones with unmet dependencies: config, feature, etc.
  - Default kselftest run attempts to run all tests skipping the ones with unmet dependencies.

Discussion - Q&A